Statistical Techniques in Robotics (16-831, S21) Lecture #10 (Monday, March 8)

Online Techniques for Supervised Learning

Lecturer: Kris Kitani *Scribes: Xuhua Huang, Xinjie Yao*

1 Review

In last two lectures, we start to focus on the Online Convex Optimization problem. We will give a brief recap for 3 representative algorithms (OMD, OGD, ONEGD) in this section. Then we will move on to Online Supervised Learning, and introduce two online techniques (Hard-SVM and Soft-SVM) to solve Supervised Learning problem in the online schema.

1.1 Online Mirror Descent (OMD)

Online Mirror Descent (OMD) is a special instance of Follow the Regularized Leader (FTRL) by applying linear loss and convex regularizer on FTRL. It can also be viewed as a generic form of FTRL with linear loss, by changing some notations and introducing the concept of dual space and primal space:

- $\boldsymbol{z}^{(1:t)} = \sum_{i=1}^{t} \boldsymbol{z}^{(i)}$ (sum of gradients)
- $\boldsymbol{\theta}^{(t+1)} \triangleq -\boldsymbol{z}^{(1:t)}$ (denote $\boldsymbol{\theta}$ as the sum of gradients)
- $\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \boldsymbol{z}^{(t)}$ (denote $\boldsymbol{\theta}$ as a incremental update from last timestep)
- $\boldsymbol{w}^{t+1} = g\left(\boldsymbol{\theta}^{(t+1)}\right) \triangleq \arg\max_{\boldsymbol{w}} \left\langle \boldsymbol{w}, \boldsymbol{\theta}^{(t+1)} \right\rangle - \psi(\boldsymbol{w})$ (mirror function from dual to primal space)

With this new notation, we can now write down the pseudo-code of OMD in Algorithm 1.

Algorithm 1 Online Mirror Descent (Convex set $S, g : \mathbb{R}^D \to S$)

1: **for** $t = 1, \cdots, T$ **do**
2: RECEIVE ($\mathbf{f}^{(t)} : S \to R$) ▷ Receive loss function
3: $\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \eta \boldsymbol{z}^{(t)}, \boldsymbol{z} \in \partial f^{(t)}\left(\boldsymbol{w}^{(t)}\right)$ ▷ Dual parameter update
4: $\boldsymbol{w}^{(t+1)} \leftarrow g\left(\boldsymbol{\theta}^{(t+1)}\right)$ ▷ Mirror projection
5: **end for**

OMD is an algorithm trying to solve optimization in dual space, and mirror the output back to primal space. The choice of different regularization functions can lead to different dual spaces.

With the help of Convex Conjugate function and Bregman Divergence, we can derive the general regret bound of OMD as:

$$R(\boldsymbol{u}) = \sum_{t=1}^{T} \boldsymbol{w}^{(t)} \cdot \boldsymbol{z}^{(t)} - \boldsymbol{u} \cdot \boldsymbol{z}^{(t)} \quad \leq \psi(\boldsymbol{u}) - \psi\left(\boldsymbol{w}^{(1)}\right) + \sum_{t=1}^{T} D_{\psi^*}\left(-\boldsymbol{z}^{(1:t)} \| -\boldsymbol{z}^{(1:t-1)}\right) \tag{1}$$

ψ is the convex regularization function, and D_{ψ^*} is the Bregman Divergence when taking the convex conjugate of the regularizer. From Equation 1, we notice that choosing a different regularization function ψ changes the regret bound and leads to a different algorithm.

1.2 Online Gradient Descent (OGD)

Gradient descent is a standard approach for minimizing differentiable convex functions. Three perspectives are introduced to understand the rationality behind.

Geometric: An intuitive way of finding the minima of f is to move in the direction opposite of the gradient. Given a convex and differentiable function, we could approach the minima using Algorithm 2.

Algorithm 2 Gradient Descent (f)

1: $\mathbf{w}^{(0)} \leftarrow \mathbf{0}$
2: **for** $t = 1, \cdots, T$ **do**
3: COMPUTE($\nabla f(\mathbf{w}^{(t-1)})$)
4: $\mathbf{w}^{(t)} = \mathbf{w}^{(t-1)} - \eta \nabla f(\mathbf{w}^{(t-1)})$
5: **end for**

Linear approximation with regularization: A more rigorous way is to bound this convex function f by its Taylor series approximation at $\mathbf{w}$.

$$f(\mathbf{u}) \geq f(\mathbf{w}) + \langle \mathbf{u} - \mathbf{w}, \nabla f(\mathbf{w}) \rangle$$

However, minimizing the RHS directly yields a solution close to negative infinity. Because Taylor series expansion is only accurate around $\mathbf{w}$, an additional constraint is included that minimizes the distance between $\mathbf{u}$ and $\mathbf{w}$ with the squared L2 norm, i.e. $\min_{\mathbf{w}} \|\mathbf{u} - \mathbf{w}\|_2^2$. Then, the objective function becomes a linear loss with quadratic regularization.

$$\mathbf{w}^{(t+1)} = \arg\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w} - \mathbf{w}^{(t)}\|^2 + \eta\Big(f(\mathbf{w}^{(t)}) + \langle \mathbf{w} - \mathbf{w}^{(t)}, \nabla f(\mathbf{w}^{(t)}) \rangle\Big)$$

Isometric quadratic approximation: By isometric quadratic approximation, we can further derive its lower bound as

$$f(\mathbf{u}) \approx f(\mathbf{w}) + (\mathbf{u} - \mathbf{w})^T \nabla f(\mathbf{w}) + \frac{1}{2\eta}(\mathbf{u} - \mathbf{w})^T \boldsymbol{I} (\mathbf{u} - \mathbf{w}),$$

It turns out to be the same objective function as the second perspective's. Then we simply use $\mathbf{w}^{(t)} = \mathbf{w}^{(t-1)} - \eta \nabla f(\mathbf{w}^{(t-1)})$ to locate its minima, which is based on a **quadratic approximation** of the loss function f. Next, we show the regret bound of online gradient descent is

$$R_{OGD} \leq DG\sqrt{T}$$

where

$$D = \max ||\mathbf{u}||_2 \quad \mathbf{u} \in S$$
$$G = \max ||\mathbf{z}||_2 \quad \mathbf{z} \in \partial f(\mathbf{w})$$

1.3 Online Normalized Exponentiated Gradient Descent (ONEGD)

We have already known different regularizers can lead to different algorithms. If we stick to linear loss function, but define regularization function as:

$$\psi(\boldsymbol{w}) = \sum_{k=1}^{K} w_k \log w_k \quad \boldsymbol{w} \in \mathbb{S}^K \tag{2}$$

We can have a new update rule as:

$$\boldsymbol{w}^{(1+t)} = \arg\min_{\boldsymbol{w}} \left\langle \boldsymbol{w}, -\boldsymbol{\theta}^{(t+1)} \right\rangle + \sum_{k=1}^{K} w_k \log w_k + \lambda \left(1 - \sum_k w_k\right) \tag{3}$$

The extra item at the end is because using entropy regularization requires the weights sum to 1. To solve for the minimizer, we just need to take derivative of following equation:

$$\mathcal{L} = \left\langle \boldsymbol{w}, -\boldsymbol{\theta}^{(t+1)} \right\rangle + \frac{1}{\eta} \sum_{k=1}^{K} w_k \log w_k + \lambda \left(1 - \sum_k w_k\right) \tag{4}$$

and get the minimizer:

$$w_n = \frac{\exp\left(\eta\theta_k\right)}{\exp(1-\eta\lambda)} \tag{5}$$

which can be viewed as a mirror projection from dual space to primal space.

Algorithm 3 Online Norm-Exp-GD (η)

1: **for** $t = 1, \cdots, T$ **do**
2: $\quad \boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \eta \boldsymbol{z}^{(t)}, \quad \boldsymbol{z}^{(t)} \in \partial f^{(t)}\left(\boldsymbol{w}^{(t)}\right)$ ▷ Dual parameter update
3: $\quad \boldsymbol{w}^{(t+1)} \propto \exp\left(\eta \boldsymbol{\theta}^{(t+1)}\right)$ ▷ Mirror projection
4: **end for**

2 Summary

Before we dive into the algorithms for Online Supervised Learning, it is better to connect it with Online Learning and Supervised Learning by Figure 1. Though some supervised learning problems will perform a batch learning and only test after training the algorithm offline, there are still some scenarios where online techniques would be more suitable. For example, there are tons of Go playing strategies for AlphaGo to learn from, it is not reasonable to test its performance and update parameters after going through all data. Therefore, we may introduce the Online Learning schema (training + testing loop) into Supervised Learning, which is Online Supervised Learning we are going to discuss.

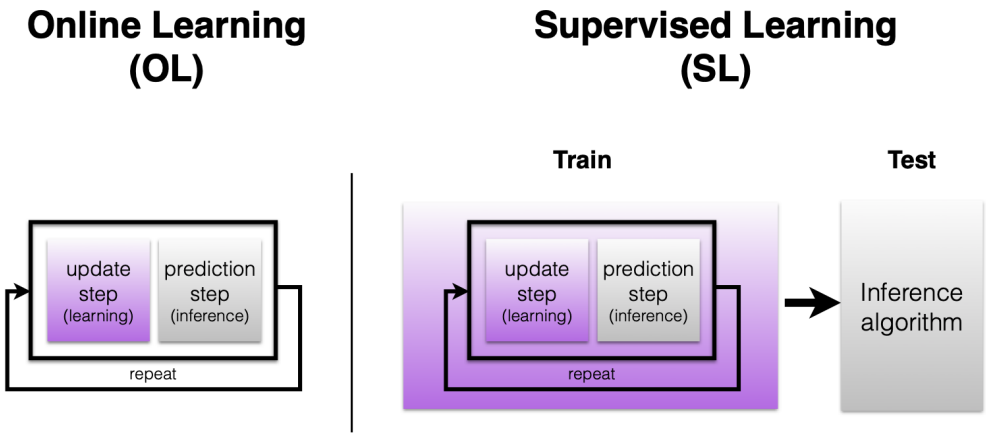

Figure 1: Online Learning vs. (Online) Supervised Learning

2.1 Hyperplane

First we need to review some fundamental concepts about Hyperplane.

2.1.1 Hyperplane in 2D

As shown in Figure 2, in 2D, hyperplane is a line denoted as $\mathbf{w} \cdot \mathbf{x} + b = 0$, where $\mathbf{w}$ and $\mathbf{x}$ are 2×1 vector. We can also notice that scaling $\mathbf{w}$ by any factor will still give a same hyperplane.

Lemma 1. *The distance from a point to a line can be written as:*

$$\text{distance}\left(w_1 x + w_2 y + b = 0, (x_0, y_0)\right) = \frac{|w_1 x_0 + w_2 y_0 + b|}{\sqrt{w_1^2 + w_2^2}} \tag{6}$$

Therefore, by plugging in the origin point $(0, 0)$, we can have distance $\rho = \frac{b}{\|\boldsymbol{w}\|}$. When we have two hyperplane $\mathbf{w} \cdot \mathbf{x} + b = 0$ and $\mathbf{w} \cdot \mathbf{x} + b = -1$, their difference of distance from origin will be:

$$\frac{b+1}{\|\boldsymbol{w}\|} - \frac{b}{\|\boldsymbol{w}\|} = \frac{1}{\|\boldsymbol{w}\|} \tag{7}$$

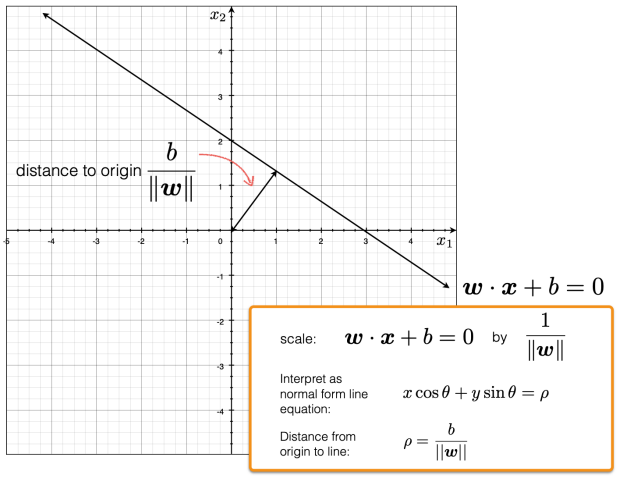

Figure 2: Illustration of Hyperplane in 2D and its distance from origin

2.1.2 Hyperplane in 3D

The distance formula still holds in 3D coordinate. The distance indicated by the blue arrow in Figure 3 can be derived by:

$$\frac{b+1}{\|\boldsymbol{w}\|} - \frac{b-1}{\|\boldsymbol{w}\|} = \frac{2}{\|\boldsymbol{w}\|} \tag{8}$$

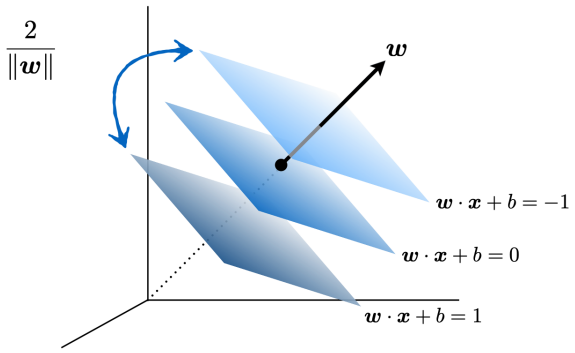

Figure 3: Illustration of Hyperplane in 3D and distance between two hyperplanes

2.2 Support Vector Machine (SVM)

2.2.1 Margin

Definition 2. *Margin is the distance from the decision hyperplane to the closest data point (on both sides).*

The best hyperplane should be maximally far away from any data point. As visualized in Figure 4, to find an optimal **w**, we must find a hyperplane which can maximize the distance between two

parallel hyperplanes (i.e. $\frac{2}{\|\mathbf{w}\|}$).

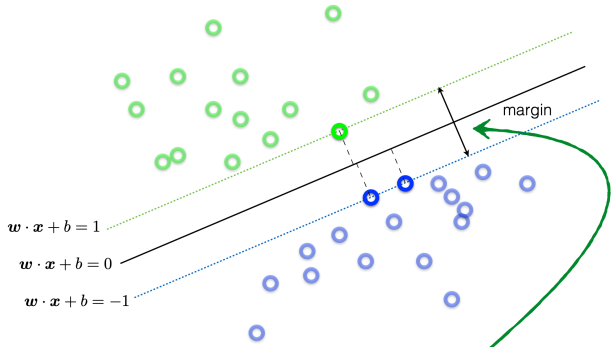

Figure 4: Illustration of optimal margin

2.2.2 Hard Margin SVM

Based on the definition in Section 2.2.1, we can now define Linear SVM as a algorithm to find a hyperplane maximizing the margin:

$$\begin{aligned} &\max_{\boldsymbol{w}} \frac{2}{\|\boldsymbol{w}\|} \\ \text{subject to } \boldsymbol{w} \cdot \boldsymbol{x}_i + b &\begin{cases} \geq +1 \text{ if } y_i = +1 \\ \leq -1 \text{ if } y_i = -1 \end{cases} \quad \text{for } i = 1, \ldots, N \end{aligned} \tag{9}$$

To make it as a convex quadratic programming (QP) problem, we can convert the objective function as:

$$\begin{aligned} &\min_{\boldsymbol{w}} \|\boldsymbol{w}\|^2 \\ &\text{subject to } y_i \left(\boldsymbol{w} \cdot \boldsymbol{x}_i + b\right) \geq 1 \text{ for } i = 1, \ldots, N \end{aligned} \tag{10}$$

where $y_i \left(\boldsymbol{w} \cdot \boldsymbol{x}_i + b\right) \geq 1$ is just a restatement of constraint in Equation 9.

Note 3. *Equation 10 is a maximization problem with hard constraints, by enforcing correct classification of all points.*

2.3 Soft Margin SVM

In real world setting, the training data can be noisy or even non-separable, which makes it impossible to find a hyperplane perfectly classifies the data. Therefore, we should allow certain degree of mistakes in order to avoid overfitting and to make our decision hyperplane more robust. This leads to the soft margin version of SVM, by adding a positive slack variable ξ_i in the criterion:

$$y_i \left(\boldsymbol{w}^\top \boldsymbol{x}_i + b\right) \geq 1 - \xi_i \tag{11}$$

We can treat ξ_i as the distance from the misclassified point to its corresponding margin. Intuitively, if its value becomes larger, it will allow more mistakes and the constraint will become softer.

With this new constraint, we can change our objective to:

$$\begin{aligned} &\min_{\boldsymbol{w}, \boldsymbol{\xi}} \|\boldsymbol{w}\|^2 + C \textstyle\sum_i \xi_i \\ &\text{subject to } y_i \left(\boldsymbol{w}^\top \boldsymbol{x}_i + b\right) \geq 1 - \xi_i \text{ for } i = 1, \ldots, N \end{aligned} \tag{12}$$

where we allow a slack variable in the constraint but still want to keep it as small as possible. C is a regularization hyperparameter, if we apply a small C, it will ignore the value of ξ, which basically allows mislabeling every data label, and generate a larger margin (soft margin). On the contrary, with a large C, we emphasize more on constraints and make the margin smaller (hard margin).

One way to formulate the objective function is as,

$$\min_{\mathbf{w}} \left(\frac{\lambda}{2}||\mathbf{w}||^2 + \frac{1}{M}\sum_{m=1}^{M}(1 - y_m \mathbf{w}^T \mathbf{x}_m) \right) \tag{13}$$

However, the slack value $(1 - y_m \mathbf{w}^T \mathbf{x}_m)$ does not perform as what we want. It causes correct points with large negative values over-weigh the misclassified points with small values. Instead, it is supposed to penalize for misclassified points or weakly correct points (ignoring the very correct points). By replacing the slack value with a hinge loss, we achieve a convex and non-differentiable optimization function.

$$\min_{\mathbf{w}} \left(\frac{\lambda}{2}||\mathbf{w}||^2 + \frac{1}{M}\sum_{m=1}^{M}\max\{0, 1 - y_m \mathbf{w}^T \mathbf{x}_m\} \right) \tag{14}$$

2.3.1 Online sub-gradient descent

Lemma 4. *Gradient descent requires a convex and differentiable loss function.*

Definition 5. *A **sub-gradient** is any $g \in \Re^n$ (same dimension as x) such that:*

$$f(y) \geq f(x) + g^T(y - x), \ \forall y$$

Definition 6. *Set of sub-gradients at $x = w$ is the **differential set** for $f(w)$.*

As the loss function is convex but non-differentiable, we will apply sub-gradient descent to find the minima according to lemma 4. Online sub-gradient descent uses any one from the set of sub-gradients at the non-differentiable points.

We consider two possible cases for sub-gradients of the hinge loss. One is when data points are correctly classified, sub-gradient z_m is set to 0. The other is when data points lie within the margin or are misclassified, sub-gradient is set to $-y_m \mathbf{x}_m$.

$$\mathbf{z}_m = \begin{cases} \mathbf{0} & \text{if } y_m \mathbf{w}^T \mathbf{x}_m \geq 1, \\ -y_m \mathbf{x}_m & \text{otherwise} \end{cases}$$

Then we can write the algorithm in online mirror descent form as Algorithm 4.

Algorithm 4 SoftSVM(λ)

1: $\boldsymbol{\theta}^{(1)} \leftarrow \mathbf{0} \in \mathbb{R}^N$
2: **for** $t = 1, ..., T$ **do**
3: $\quad \boldsymbol{x_d}, y_d \sim \boldsymbol{D}$ ▷ Receive sample from environment
4: $\quad \boldsymbol{\theta}^{(t)} = \boldsymbol{\theta}^{(t-1)} + y_d \boldsymbol{x_d} \cdot \mathbf{1}[y_d(\boldsymbol{w}^{(t)} \cdot \boldsymbol{x_d}) < 1]$ ▷ Dual parameter update
5: $\quad \boldsymbol{w}^{(t+1)} \leftarrow \frac{1}{\lambda(t+1)} \boldsymbol{\theta}^{(t)}$ ▷ Mirror Projection
6: **end for**
7: $\bar{\boldsymbol{w}} = \frac{1}{T}\sum_t \boldsymbol{w^{(t)}}$

2.3.2 Comparison with Perceptron

Recall Perceptron algorithm written in online mirror descent form as Algorithm 5,

Algorithm 5 Perceptron

$\boldsymbol{w}^{(1)} \leftarrow 0$
for t=1,...,T **do**
  RECEIVE $(\boldsymbol{x}^{(t)}, y^{(t)}) \in \mathbb{R}^N$ ▷ Receive sample from environment
  $\boldsymbol{\theta}^{(t)} = \boldsymbol{\theta}^{(t-1)} + y^{(t)}\boldsymbol{x}^{(t)} \cdot \mathbf{1}[y^{(t)}\langle \boldsymbol{w}^{(t)}, \boldsymbol{x}^{(t)}\rangle < 0]$ ▷ Dual parameter update
  $\boldsymbol{w}^{(t+1)} \leftarrow \boldsymbol{\theta}^{(t)}$ ▷ Mirror Projection
end for

Observation 7. *There are some similarities and differences between Perceptron algorithm and SoftSVM algorithm.*

1. *Both of them have piece-wise linear loss with quadratic regularization.*
2. *Dual parameter update diverges: SoftSVM allows a soft margin while Perceptron uses no margin.*
3. *Mirror function differs: SoftSVM has a multiplication update while Perceptron uses an identity mapping (step size could be applied to Perceptron).*

2.4 Conclusion

In this lecture, we dive into the Online Supervised Learning problem, by introducing a representative algorithm called Support Vector Machine (SVM). We turn it into an online learning algorithm by introducing the hinge loss and sub-gradient descent. To handle non-separable or noisy data, we allow it to have a soft margin, aiming to make it more robust in testing environment. SVM can be written in the form of OMD. Interestingly, it is very similar to Perceptron while they are slightly different in the dual parameter update and mirror function.

3 Appendix

3.1 More on Online SVM

Under the online learning setting, SVM is continuously being trained with real-time data input coming in. One of the most popular online SVM solver is called **LASVM** proposed in [2]. LASVM is used for big data stream mining, and was introduced by Bordes in 2005. Figure 5 shows the overall pipeline for LASVM, please refer to the paper to get more details. The basic idea is to incorporate SVM with online kernel classifiers. The algorithm uses the traditional SVM (Quadratic Programming) solver with online kernel approximation which leverages a similar single sequential pass method used in SVM [1].

LASVM

1) **Initialization**:
   Seed $\mathcal{S}$ with a few examples of each class.
   Set $\alpha \leftarrow 0$ and compute the initial gradient g (equation 9)
2) **Online Iterations**:
   Repeat a predefined number of times:
   - Pick an example k_t
   - Run PROCESS(k_t).
   - Run REPROCESS once.
3) **Finishing**:
   Repeat REPROCESS until $\delta \leq \tau$.

Figure 5: LASVM algorithm

3.2 Comparison between online SVM and offline SVM

There are two insightful tables for comparing online SVM and offline SVM from [1].

In terms of the data types, constant and small data set size may be more suitable for offline SVM, whereas a continuously changing data will be most suitable to online SVM. Online SVM can handle stationary and non-stationary data where it is continuously changing its patterns.

Data Types

Offline SVM	Features	Online SVM
Fixed sample of data that is passed once into the model	Data Input	Continuous and real time data that passes many times to the model
Noisy data can cause performance issues with outliers	Noisy Data	Handles noisy data quite well with adaptive hyperplane
Cannot handle nonstationary data	Data Handling	Handles both stationary and nonstationary data well

Figure 6: Comparison in data types

In terms of model features, online SVM is less complex in handling data of different patterns due to its ability to re-train the algorithm when a new sample is given. It is able to adapt to data input stream and to adjust accordingly.

Model Features

Offline SVM	Features	Online SVM
Complexity increases as the number of data or there is a shift of data pattern	Complexity	Less complex due to its ability to learn different data patterns and types
Cannot handle drift in data. Performance will degrade	Concept Drift	Can handle drift and re-train the model to adapt
Linear and Non-Linear	Classifier	Linear and Non-Linear
Fixed with the sample of data it is trained	Hyperplane	Adaptive to the data input and adjust accordingly

Figure 7: Comparison in model features

3.3 Other good matrials for SVM / Online SVM

- Introduction to Support Vector Machine: http://cs229.stanford.edu/notes/cs229-notes3.pdf
- Support Vector Machines for Classification: *https://www.ugpti.org/smartse/resources/downloads/support-vector-machines.pdf*
- Linear and Non Linear Classifier: *https://towardsdatascience.com/understanding-support-vector-machine-part-2-kernel-trick-mercers-theorem-e1e6848c6c4d*
- Active Support Vector Machine: *http://sdep.cs.aueb.gr/docs/Vlachos-Active-Learning-MScThesis.pdf*

References

[1] Big data stream mining with online support vector machines, Sep 2020.

[2] J. W. Antoine Bordes, Seyda Ertekin. Fast kernel classifiers with online and active learning. *Journal of Machine Learning Research*, 2005.